MULTI-VALUED CONTROL PROBLEMS AND MIXTURE DENSITY NETWORK

Randa Herzallah and David Lowe
email: herzarom, d.lowe@aston.ac.uk

NCRG, Aston University, UK

Abstract: We have proposed a novel robust inversion-based neurocontroller that searches for the optimal control law by sampling from the estimated Gaussian distribution of the inverse plant model. However, for problems involving the prediction of continuous variables, a Gaussian model approximation provides only a very limited description of the properties of the inverse model. This is usually the case for problems in which the mapping to be learned is multi-valued or involves hysteritic transfer characteristics. This often arises in the solution of inverse plant models. In order to obtain a complete description of the inverse model, a more general multicomponent distributions must be modeled. In this paper we test whether our proposed sampling approach can be used when considering an arbitrary conditional probability distributions. These arbitrary distributions will be modeled by a mixture density network. Importance sampling provides a structured and principled approach to constrain the complexity of the search space for the ideal control law. The effectiveness of the importance sampling from an arbitrary conditional probability distribution will be demonstrated using a simple single input single output static nonlinear system with hysteretic characteristics in the inverse plant model.

Keywords: Uncertainty. Neural networks. Sampling. Nonlinear Processes. Mixture density network. Conditional distributions.

1. INTRODUCTION

Standard inverse controllers based on a least square approach lead to extremely poor performance when applied to inverse problems in which the mapping to be learned is multi-valued or involves hysteretic transfer characteristics (Nabney, 2002). This is due to the fact that when a least square approach is applied to an inverse problem, it will then approximate the conditional average of the target data. However, the average of several solutions is not necessarily a correct solution. In (Nabney, 2002; Evans and Cornford, 2000) a new class of network models obtained by combining a conventional neural network with a mixture density model, has been used to model the conditional probability distribution for problems in which the mapping to be learned is multi-valued. Other computational approaches, namely forward and inverse modelling, and feedback error learning have been suggested in (Sutton and Werbos, 1990; White and Sofge, 1992) for acquiring the inverse dynamics model of multi-valued functions. However the output from the above mentioned approaches has been an estimation for the control value only. Although a mixture density network models the conditional probability distribution, it uses only a single control value when used as a controller in the control loop. This value is the mean value of one of the kernel functions corresponding to the most probable branch. Recently growing interest in robust control by accounting for model and system uncertainty has produced new results. For example, in (Ayala Botto *et al.*, 2000) a systematic procedure that accounts for the structured un-

certainty when a neural network model is integrated in an approximate feedback linearisation control scheme has been developed. A different way for accounting for the uncertainty around the predicted output of the inverse controller has been presented in (Herzallah and Lowe, 2002*b*; Herzallah and Lowe, 2002*a*). The controller is designed to predict both the control law and the uncertainty around that control law, which leads to the assumption that the inverse controller can be approximated by a Gaussian function. A sampling approach is used to search for a better value of the control signal than the mean in this region where the optimal solution is expected to lie. The stability for the updating rule of the control law has been proved in (Herzallah and Lowe, 2002*a*). However the Gaussian assumption is not always possible, as for example problems where the inverse mapping can be multi-valued. This paper will go beyond the Gaussian description of the distribution of the inverse controller. The main idea is to use the mixture density network to model the multicomponent distributions of the inverse model of the plant. The idea of the mixture density network is not new (Nabney, 2002; Evans and Cornford, 2000), but it has not been exploited in a control context before. The work presented here differs from (Nabney, 2002) in that we consider the multi-component distribution to search for the optimal control law, rather than taking a single estimate value corresponding to the most probable value. In (Herzallah and Lowe, 2002*b*; Herzallah and Lowe, 2002*a*) only the Gaussian distribution is considered. We extend this work by considering more general distributions, which create a general framework for searching for the optimal control law from an arbitrary probability distribution.

2. MIXTURE DENSITY NETWORK

In standard inverse control the challenge is to build a neural network that will take past values of the input, u and output, y of the plant $x(t) = [y(t-1), ..., y(t-n), u(t-d-1), ..., u(t-n)]$ and the desired output value $y_r(t)$ as an input, and outputs the control signals $u(t-d)$ (assuming d relative degree, and n is the known plant order), which will move the plant output to the desired value. In this work the basic goal is to model the statistical properties of the control signals, $u(t-d)$, expressed in terms of the conditional distribution function $p(u(t-d)|s(t))$. Here $s(t) = [x(t), y_r(t)]$ is the input vector to the neural inverse model. For dynamical systems it is reasonable to assume that the output of the system $y(t)$ is function f of its input $u(t-d)$ and the delayed vector $x(t)$. Furthermore in the case of a one-to-one mapping, and only in this case, the inverse of the function denoted by f^{-1} can be solved by minimisation of a sum of squares error function. For multi-valued functions, Mixture Density Networks (MDNs) (Nabney, 2002) provide a general framework for modelling conditional probability density functions $p(u(t-d)|s(t))$ for the inverse mapping. The distribution of the outputs, $u(t-d)$, is described by a parametric model whose parameters are determined by the output of a neural network, which takes $s(t)$ as inputs. The general conditional distribution function is given by

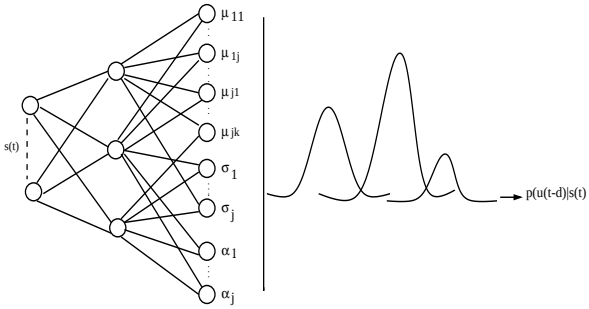

Fig. 1. The structure of a Mixture Density Network.

$$p(u(t-d)|s(t)) = \sum_{j=1}^{M} \alpha_j(s(t))\phi_j(u(t-d)|s(t)) \quad (1)$$

where $\alpha_j(s(t))$ represents the mixing coefficients, and can be regarded as prior probabilities (which depend on $s(t)$), $\phi_j(u(t-d)|s(t))$ are the kernel distributions of the mixture model (whose parameters are also conditioned on $s(t)$), and M is the number of kernels in the mixture model. Various choices are available for the kernel functions, but in this paper the choice will be restricted to spherical Gaussians of the form

$$\phi_j(u(t-d)|s(t)) = \quad (2)$$
$$\frac{1}{(2\pi)^{c/2}\sigma_j^c(s(t))} \exp\left(-\frac{\| u(t-d) - \mu_j(s(t)) \|^2}{2\sigma_j^2(s(t))}\right)$$

where c is the dimensionality of the target data $u(t-d)$, $\mu_j(s(t))$ represents the centre of the jth kernel, with components μ_{jk}. A spherical Gaussian assumption can be relaxed in a very straightforward way, by using full covariance matrices for each Gaussian kernel. However this complication is not necessary, because in principle a Gaussian Mixture Model (GMM) with sufficiently many kernels of the type given by (2) can approximate any given density function arbitrarily accurately providing that the mixing coefficients and the Gaussian parameters are correctly chosen (Nabney, 2002). It follows then that for any given value of $s(t)$, the mixture model (1) provides a general formalism for modelling the conditional density function $p(u(t-d)|s(t))$. To achieve this the parameters of the mixture model, namely the mixing coefficients $\alpha_j(s(t))$, the means $\mu_j(s(t))$ and the variance $\sigma_j^2(s(t))$ are taken to be general continuous functions of $s(t)$. These functions are modelled by the outputs of a feed-forward neural network that takes $s(t)$ as input. This combination of a GMM whose parameters dependent on the output of a feed forward neural network that takes $s(t)$ as an input is refered to as an MDN and is represented schematically in figure 1. The neural network element of the (MDN) is implemented

with a standard radial basis function network (RBF) of thin plate spline basis functions. The output vector from the RBF, Z, holds the parameters that define the Gaussian mixture model. For M components in the mixture model (1) the network will have $(c+2) \times M$ outputs, namely M outputs denoted by z_j^α which determines the mixing coefficients α_j, M outputs denoted by z_j^σ which determine the kernel width σ_j, and $M \times c$ outputs denoted by z_{jk}^μ which determine the components μ_{jk} of the kernel centres μ_j. This is compared with the usual c outputs for a RBF network used with a sum-of squares error function. The outputs of the MDN undergo some transformations to satisfy the constraints of the mixture model. The constraints are such that

$$\sum_{j=1}^{M} \alpha_j(s(t)) = 1, \quad 0 \leq \alpha_j(s(t)) \leq 1 \tag{3}$$

The first constraint ensures that the distribution is correctly normalised, so that $\int p(u(t-d)|s(t))\mathrm{d}u(t-d) = 1$. These constraints can be satisfied by choosing $\alpha_j(s(t))$ to be related to the network's outputs by a 'softmax' function

$$\alpha_j(s(t)) = \frac{\exp(z_j^\alpha)}{\sum_{l=1}^{M} \exp(z_l^\alpha)} \tag{4}$$

The variances of the kernel represent scale parameters and always take positive values, so it is convenient to represent them in terms of the exponentials of the corresponding outputs of the RBF network, z_j^σ

$$\sigma_j^2 = \exp(z_j^\sigma) \tag{5}$$

The centres μ_j of the Gaussians represent a location in the target space and can take any value within that space. Therefore they are taken directly from the corresponding outputs of the RBF network, z_{jk}^μ

$$\mu_{jk} = z_{jk}^\mu \tag{6}$$

In order to optimise the parameters in a MDN, an error function is required that provides an indication of how well the model represents the underlying generating function of the training data. The error function of the mixture density network is motivated from the principle of maximum likelihood (Nabney, 2002). The likelihood of the training data set, $\{s(t), u(t-d)\}$, can be written as

$$\mathcal{L} = \prod_n p(u_n(t-d)|s_n(t))p(s_n(t)) \tag{7}$$

where here the assumption has been made that each data point has been drawn independently from the same distribution, and so the likelihood is a product of probabilities. Generally one wishes to maximise the likelihood function, which is equivalent to minimising the negative logarithm of the likelihood function. The negative log likelihood can be regarded as an error function, E

$$\begin{aligned} E &= -\sum_n \ln p(u_n(t-d)|s_n(t)) - \sum_n p(s_n(t)) \quad (8) \\ &\approx -\sum_n \ln\left\{ \sum_{j=1}^{M} \alpha_j(s_n(t))\phi_j(u_n(t-d)|s_n(t)) \right\} \end{aligned}$$

where we have dropped the last (constant) term and used (1). In order to minimise the error function, the derivatives of the error E with respect to the weights in the neural networks must be calculated. Providing that the derivatives can be computed with respect to the outputs of the network, the errors at the network inputs may be calculated using the back-propagation procedure (Nabney, 2002). By first defining the posterior probability of the jth kernel, using Bayes theorem

$$\pi_j(s(t), u(t-d)) = \frac{\alpha_j \phi_j}{\sum_{l=1}^{M} \alpha_l \phi_l} \tag{9}$$

the analysis of the error derivatives with respect to the network outputs is simplified. The computation of the error can further be simplified by considering the error derivative with respect to each training pattern, n. The total error, E, is defined as a summation of the error, E_n, for each training pattern. $E = \sum_{n=1}^{N} E^n$, where

$$E^n = -\ln\left\{ \sum_{j=1}^{M} \alpha_j(s_n(t))\phi_j(u_n(t-d)|s_n(t)) \right\} \tag{10}$$

Each of the derivatives of E^n are considered with respect to the outputs of the networks and their respective labels for the mixing coefficients, z_j^α, variance parameters, z_j^σ and centres or position parameters z_{jk}^μ. The derivatives are as follows.

$$\frac{\partial E^n}{\partial z_j^\alpha} = \alpha_j - \pi_j \tag{11}$$

$$\frac{\partial E^n}{\partial z_j^\sigma} = -\frac{\pi_j}{2}\left\{ \frac{\| u_n(t-d) - \mu_j \|^2}{\sigma_j^2} - c \right\} \tag{12}$$

$$\frac{\partial E^n}{\partial z_{jk}^\mu} = \pi_j \left\{ \frac{\mu_{jk} - u_k(t-d)}{\sigma_j^2} \right\} \tag{13}$$

For full derivation see (Nabney, 2002). Once the network has been trained it can predict the conditional density function of the target data for any given value of the input vector. This conditional density represents a complete description of the generator of the data. More specific quantities can be calculated from this density function which may be of interest in different applications. An example of these quantities is the mean, corresponding to the conditional average of the target data. This corresponds to the mean computed by a standard network trained by least squares. However, in control applications where unique solutions cannot be found, and where the distribution of the target data will consist of different numbers of distinct branches, this is a not valid solution. In such cases one may be interested in finding an output value corresponding to the most probable branch. Since each component of the mixture model is normalised, $\int \phi_j(u(t-d)|s(t))\mathrm{d}u(t-d) = 1$, the most probable branch is

given by $\arg\max_j\{\alpha_j(s(t))\}$ The required value of $u(t-d)$ is then given by the corresponding centre μ_j.

3. INCORPORATING UNCERTAINTY FOR THE MIXTURE DENSITY NETWORK

Since the proposed sampling algorithm for the Gaussian function has been covered in (Herzallah and Lowe, 2002*b*; Herzallah and Lowe, 2002*a*), we summarise here the main steps to apply the same algorithm considering sampling from a more general distribution. The architecture of this algorithm is shown in figure 2.

(1) An accurate model of the process needs to be constructed based on the pre-collected input-output data, and to be trained off line. In the general case, it is assumed to be described by the following neural network model:

$$\hat{y}(t) = N_f(x(t), u(t-d)) \tag{14}$$

(2) The conditional distribution of the inverse model of the plant should also be constructed. It is assumed to be described by a mixture density network given by equation (1).

(3) For the non-sampling case, in the mixture density network the value of the control signal is assumed to be given by the centre μ_j of the most probable branch, where the most probable branch is given by

$$\arg\max_j\{\alpha_j(s(t))\} \tag{15}$$

(4) for the sampling approach the following steps need to be carried out at each instant of time, t:
   (a) The desired output is calculated from the reference model output, which should be chosen to have the same relative degree as that of the plant.
   (b) Calculate the components μ_{jk} of the kernel centres μ_j, and the kernel width σ_j of each kernel function, based on the desired output value.
   (c) The admissible values of the control signal for the mixture density network, are then assumed to be sampled from a mixture density network. Since we are using Gaussian kernel functions the samples can be generated from each kernel function randomly using the retrieved components μ_{jk} of the kernel centres μ_j, and the kernel width σ_j of each kernel function. The number of samples from each component is determined randomly with more samples generated from the component with larger prior.
   (d) Based on the effect of each sample on the output of the model, the most likely control value is taken, which is assumed to be the value that minimises the following cost function.

$$J(t) = \underset{u\in U}{Min} \underset{\bar{v}}{E}[(\hat{y}(t) - y_r(t))^2] \tag{16}$$

where U is a vector containing the sampled values from the control signal distribution, E is the expected value of the cost function over the random noise variable $\bar{v}$. Because we are using a neural network to model the system, and because the neural network predicts the mean value for the output of the model averaged over the noise on the data, the above function can be optimised directly.

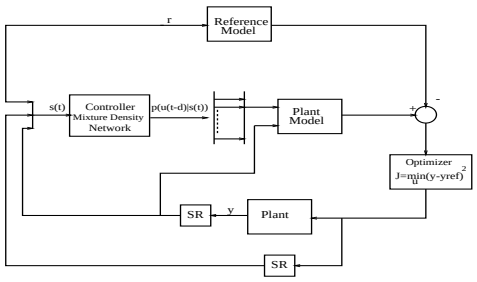

Fig. 2. The architecture of the proposed optimisation method.

For details of the stability analysis see (Herzallah and Lowe, 2002*a*).

4. SIMULATION STUDY

4.1 Introduction

For inverse problems, the mapping can often be multi-valued and a unique solution cannot be found. If the Gaussian distribution approximates the inverse model, it will approximate the conditional average of the target data, and this will frequently lead to extremely poor performance. Here we will overcome this problem by appropriate use of a Mixture Density Network instead. In order to illustrate the application of the MDN with the proposed control approach we consider a simple example of single input single output given by the following equation

$$y = u + 0.3\sin(2\pi u) + \epsilon \tag{17}$$

where ϵ is a random variable with uniform distribution in the interval $(-0.1, 0.1)$, y is the output variable, and u is the input variable. This example has been used in (Nabney, 2002; Evans and Cornford, 2000) to demonstrate the use of the Mixture Density Network. This equation represents a static system, since no delay exists between the input and the output variable. The plant has been considered to be given by equation (17). In order to identify the plant, an input-output model described by $\hat{y} = f(u)$ was chosen, where f is a thin plate spline radial basis function network. Figure 3 shows a data set of 300 points generated by sampling equation (17). Also shown is the mapping represented by a thin plate spline radial basis function

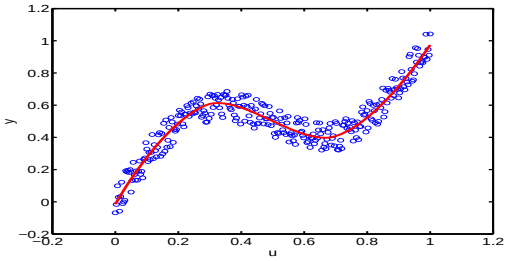

Fig. 3. The forward model of the function $y = u + 0.3\sin(2\pi u) + \epsilon$. The circles represent the samples generated from that function. The solid curve shows the result of training a thin plate spline radial basis function with 5 basis functions using a sum of square error function.

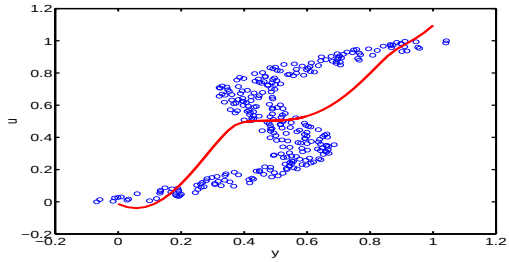

Fig. 4. The inverse model of the function $y = u + 0.3\sin(2\pi u) + \epsilon$. The circles represent the same data as in 3. The solid curve shows the result of training a thin plate spline radial basis function with 15 basis functions using a sum of square error function.

network after training using this data. The optimal structure for the neural network found by applying the cross validation method consisted of 5 thin plate spline basis functions. It was trained using the scaled conjugate gradient method. It can be seen that the network which is approximating the conditional average of the target data, gives an excellent representation of the underlying generator of the data.

4.2 Standard Inverse Control

We consider acquiring the inverse mapping of the same problem and using the same training data as in the forward model by training a thin plate spline radial basis function network using least squares. Similarly an input-output model described by $\hat{u} = f^{-1}(y)$ was chosen to find the inverse model of the plant. The network tries to approximate the conditional average of the target data, but this corresponds to a very poor representation of the process as can be seen from figure 4. The network in this case had 15 thin plate spline basis functions and was trained using the scaled conjugate gradient optimisation method. This network has been connected in series with the plant to generate the control signal required to cause the plant to follow the desired output. The desired output has been considered to be given by $y_r = r + 0.3\sin(2\pi r)$, where the input r has been chosen in such a way to generate data that have not been used in the training stage. The result is shown in figure 5, where it can be seen that there is a large error between the desired output and the plant output.

4.3 Mixture Density Network

In this section we apply an MDN to the same inverse problem, using the same data set as before. The appropriate number of kernel functions and the complexity of the neural network has been decided by applying the cross validation method. It was found that the best structure for the MDN consists of 7 thin plate spline basis functions with 9 outputs corresponding to

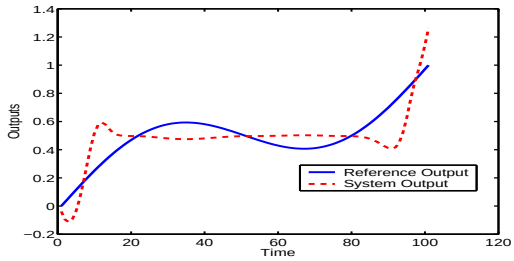

Fig. 5. The control result from using the classical inverse controller.

3 kernel functions . The MDN was trained using the scaled conjugate gradient optimisation method. Once trained the MDN predicts the conditional probability density of the target data (regarded as the input to the plant u in the inverse model) for each value of the input to the network (regarded as the output to the plant y in the inverse model). Having obtained a good representation for the conditional density of the target data, we can in principle calculate any desired statistics from that distribution. In our control problem, since the conditional mean of the target data is a very poor result, we are interested in the evaluation of the centre of the most probable kernel according to equation (15), which gives the result shown in figure 6. Again this network has been connected in series with the plant to generate the control signal required to cause the plant to follow the same desired output as before. The result is shown in figure 7, where it can be seen that using the most probable value of the kernel functions has improved the performance of the controller significantly.

4.4 Proposed Control Approach

The final experiment that we have performed, is to sample from the control signal distribution (from the mixture density distribution). In the new proposed control approach the best control signal was found and forwarded to the plant, following the procedure presented earlier. Again the control signal was obtained from a small number of samples, typically 20 samples in this case. The overall performance of the plant un-

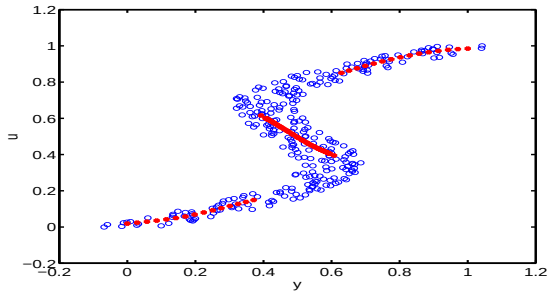

Fig. 6. Plot of the central value of the most probable kernel as a function of y from the Mixture Density Network.

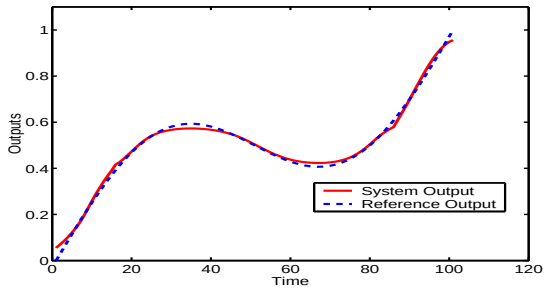

Fig. 7. The control result from using most probable value of the Mixture Density Network as a control law.

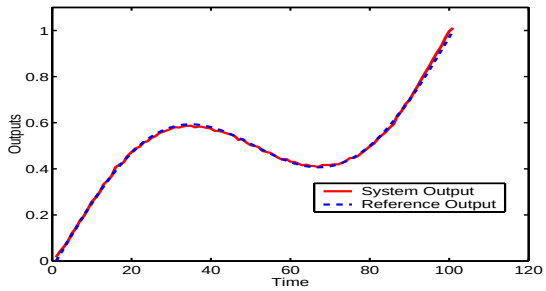

Fig. 8. The control result from applying the proposed sampling approach from the mixture density network.

der the proposed control approach is shown in figure 8. It can be seen from this figure that the proposed sampling approach performs even better than finding the most probable centre value of the kernel function. The error from the absolute difference between the plant output and the desired output of the most probable value of the kernel function in the mixture density network, and the proposed sampling approach is shown in figure 9. From this figure one can see that the sampling approach has reduced the error significantly.

5. CONCLUSIONS

General inverse control can be considered to be a good control strategy if the model of the plant happens to be invertible and accurate. The main contribution of this paper is that it extends the importance sampling approach from a Gaussian function by considering sampling from an arbitrary probability distribution func-

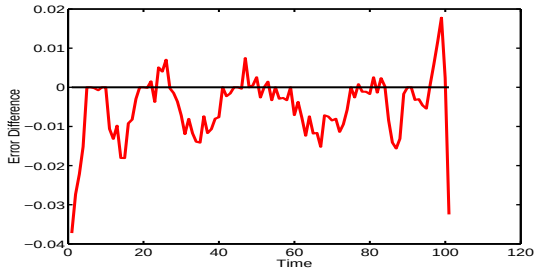

Fig. 9. The Error Difference.

tion. Simulation experiments demonstrated the successful application of the importance sampling strategy from an arbitrary probability function to improve the controller performance for a class of nonlinear single input single output static systems in which the inverse mapping is multi-valued. The example given in this paper demonstrates a whole class of density-estimating neural networks (the Mixture Density Network) and also points out a fruitful direction for control research: that of sampling control signals from estimated distribution functions which can incorporate even more information on the full distribution such as higher order moments beyond just the first two, representing the control law and the uncertainty around the control law. This more general approach is not constrained by assumptions of invertibility and it shows the ability to deal with multi-valued processes as well.

REFERENCES

Ayala Botto, Miguel, Bart Wams, Ton van den Boom and José Sá da Costa (2000). Robust stability of feedback linearised systems modelled with neural networks: dealing with uncertainty. *Engineering Applications of Artificial Intelligence* **13**(6), 659–670.

Evans, D.J., I.T. Nabney and D. Cornford (2000). Structured neural network modelling of multi-valued functions for wind vector retrieval from satellite scatterometer measurements. *Neurocomputing* **30**, 23–30.

Herzallah, R. and D. Lowe (2002*a*). Improved robust control of nonlinear stochastic systems using uncertain models. In: *Controlo2002*. Aveiro, Portugal. pp. 507–512.

Herzallah, R. and D. Lowe (2002*b*). A novel approach to modelling and exploiting uncertainty in stochastic control systems. In: *International Conference on Artificial Neural Networks, ICANN*. Madrid, Spain. pp. 801–806.

Nabney, I.T. (2002). *Netlab Algorithms for Pattern Recognition*. Springer.

Sutton, R.S., W.T. Miller and Werbos, P.J., Eds.) (1990). *Neural networks for Control*. MIT Press. Cambridge, Massachusetts, London, England.

White, D.A. and Sofge, D., Eds.) (1992). *Handbook of Intelligent Control*. Multiscience Press, Inc. New York, N.Y.